Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 727 – 736

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

A Mass Conservation Algorithm For Adaptive Unrefinement Meshes Used By Finite Element Methods

Hung V. Nguyen*[1], Jing-Ru C. Cheng[1], Charlie R. Berger[2], and Gaurav Savant[2]

[1] *U.S. Army Engineer Research and Development Center (ERDC) Information Technology Laboratory (ITL), Vicksburg, MS 39180*
email: hung.v.nguyen, ruth.c.cheng@usace.army.mil

[2] *U.S. Army Engineer Research and Development Center (ERDC) Coastal and Hydraulics Laboratory (CHL), Vicksburg, MS 39180*
email: charlie.r.berger, gaurav.savant@usace.army.mil

Abstract

The Adaptive Hydraulics (ADH) model is an adaptive finite element method to simulate three-dimensional Navier-Stokes flow, unsaturated and saturated groundwater flow, overland flow, and two- or three-dimensional shallow-water flow and transport. In the shallow-water flow and transport, especially involving multispecies transport, the water depth (h), the product of water depth and velocities (uh and vh), as well as water depth and chemical concentration (hc) are dependent variables of fluid-motion simulations and are often solved at various times. It is important for the numerical model to predict accurate water depth, velocity fields, and chemical distribution, as well as conserve mass, especially for water quality applications. Solution accuracy depends highly on mesh resolution. Adaptive mesh refinement (AMR), particularly the h-refinement, is often used to add new nodes in the region where they are needed and to remove others where they are no longer required during the simulation. The AMR is proven to optimize the performance of a computed solution. However, mass with gain or loss can occur when elements are merged due to removing a node at mesh coarsening. Therefore, we develop and implement the mass-conservative unrefinement algorithm to ensure the mass conserved in a merged element in which a node has been removed.

This study describes the use of the Galerkin finite element method to redistribute mass to nodes comprising a merged element. The algorithm was incorporated into the ADH code. This algorithm minimizes mass error during the unrefinement process to conserve mass during the simulation for two-dimensional shallow-water flow and transport. The implementation neither significantly increases the computational time nor memory usage. The simulation was run with various numbers of processors. The results showed good scaling of solution time as the number of processors increases.

Keywords: Adaptive mesh, finite element method, mass conservative, shallow-water flow, and ADH model

1. Introduction

Accurate solutions are often sought when we solve partial differential equations (PDE) for scientific problems. A variety of numerical errors (e.g., spurious oscillation, numerical spreading, grid orientation, and peak clipping/valley elevating) [1, 2, 3] often exist when we use numerical methods to discretize the PDE system. These errors can be partially mitigated through the implementation of higher order finite element or finite difference discretizations, for example, TVD and ENO methods [4, 5], with the increase of spatial and temporal resolution throughout the

doi:10.1016/j.procs.2012.04.078

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2012		**00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Mass Conservation Algorithm For Adaptive Unrefinement Meshes Used By Finite Element Methods	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army Engineer Research and Development Center (ERDC) ,Information Technolog,Vicksburg,MS,39180**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

Procedia Computer, v 9, June 2, 2012, To be presented at International Conference on Computational Science, ICCS 2012, Omaha, Nebraska, June 4-6, 2012

14. ABSTRACT

The Adaptive Hydraulics (ADH) model is an adaptive finite element method to simulate three-dimensional Navier-Stokes flow, unsaturated and saturated groundwater flow, overland flow, and two- or three-dimensional shallow-water flow and transport. In the shallow-water flow and transport, especially involving multispecies transport, the water depth (h), the product of water depth and velocities (uh and vh), as well as water depth and chemical concentration (hc) are dependent variables of fluid-motion simulations and are often solved at various times. It is important for the numerical model to predict accurate water depth, velocity fields, and chemical distribution, as well as conserve mass, especially for water quality applications. Solution accuracy depends highly on mesh resolution. Adaptive mesh refinement (AMR), particularly the h-refinement, is often used to add new nodes in the region where they are needed and to remove others where they are no longer required during the simulation. The AMR is proven to optimize the performance of a computed solution. However, mass with gain or loss can occur when elements are merged due to removing a node at mesh coarsening. Therefore, we develop and implement the mass-conservative unrefinement algorithm to ensure the mass conserved in a merged element in which a node has been removed. This study describes the use of the Galerkin finite element method to redistribute mass to nodes comprising a merged element. The algorithm was incorporated into the ADH code. This algorithm minimizes mass error during the unrefinement process to conserve mass during the simulation for two-dimensional shallow-water flow and transport. The implementation neither significantly increases the computational time nor memory usage. The simulation was run with various numbers of processors. The results showed good scaling of solution time as the number of processors increases.

15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**11**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

entire domain to avoid oscillation. But uniformly refining the computational mesh throughout the domain is way too expensive. An adaptive mesh refinement (AMR) approach, which is also called h-refinement, can be more effective in its use of computer resources for large-scale applications. The AMR improves the mesh resolution in regions where the numerical error is large, while keeping the mesh coarse elsewhere. However, the AMR technique is more complex to implement but benefits many applications exhibiting discontinuties, shock waves, or phase changes [6], etc. A wide variety of AMR approaches have been explored in the literature [7, 8, 9, 10, 11, 12], including serial and parallel approaches [13, 14, 15].

Mass conservation is the primary requirement when AMR takes place. In fact, AMR includes two parts: mesh refinement and unrefinement. During the refinement step, it is straightforward to add these nodes and elements in a manner that preserves mass, such as linear interpolation in a linear element. For unrefinement, the nodes are deleted and elements are merged in some regions where a fine mesh resolution has become useless for transient problems. The unrefinement step is necessary to avoid memory and computation overkill. Without special care of transferring data from the unrefined node to its parents, the result often brings up mass conservation issues. Yamoah [16] investigated this problem and proposed three algorithms to restore mass conservation for unrefinement when solving 1-D variably saturated groundwater flow using MATLAB code. These methods are based on either a weighted average approach or the L_2 projection approach. He concluded that the mass cannot be preserved, and the mass lost may be significant for a long simulation if a mass-conservative unrefinement algorithm is not used.

We extended the L_2 projection algorithm onto 2-D unstructured meshes. We developed the mathematical formulation and implemented the algorithm in the ADaptive Hydraulics (ADH) modeling system—a parallelized numerical model with the following components: saturated and unsaturated groundwater flow, three-dimensional Navier-Stokes flow, two- or three-dimensional shallow-water flow, and the associated transport. When refinement is on the ADH, the element is split if the solution error on the element exceeds the refinement error tolerance. At the added node the data such as water head or chemical concentration are linear interpolation in the element. When the solution error on the element is below the unrefinement error tolerance then the elements are recombined. The node is simply deleted without special care of transferring data from the unrefined node to its parents. In this paper, we first discuss the shallow-water flow application including the governing equations and discretized formulation, numerical methods, and numerical difficulties. We then present the mathematical formulation of the mass-conservative unrefinement algorithm, followed by its implementation. Experimental results are then presented to demonstrate the mass conservation throughout the entire simulation. Finally, we summarize and discuss these results and also present plans for future works.

2. ADaptive Hydraulics (ADH)

ADH is a modular, parallel, adaptive finite element model for one-, two-, and three-dimensional flow and transport. ADH simulates saturated-unsaturated groundwater flow, internal flow, and open-channel flow. The groundwater module includes solving 3-D variably saturated groundwater flow and constituent transport. The groundwater flow may be coupled to 2-D diffusive wave or dynamic wave surface water flow and can be loosely coupled to 3-D subsurface heat transport. The internal flow module is used to solve 3-D Navier-Stokes equations. This module is nonhydrostatic and includes a $\kappa - \epsilon$ turbulence model. There are two approaches for open-channel flows: nonhydrostatic and hydrostatic pressure methods. The nonhydrostatic approach is more appropriate for domains near structures where the vertical inertial terms are significant. This approach is the same as that used for the 3-D internal flow, though the free surface is updated dynamically. The hydrostatic approach is to solve the shallow-water flow equations, including 2-D and 3-D variable density flow (baroclinic). The 2-D module includes the capability for wetting-drying, dam-break, supercritical and subcritical flow, sediment transport, and bed change, and also has the capability to include the long-wave vessel effects.

The 2-D shallow-water flow equations are obtained by vertically integrating the mass and momentum under the assumptions of incompressible flow and hydrostatic pressure [17]. Assuming negligible free surface shear and negligible fluid pressure at the free surface, the 2-D shallow-water equations are written as the following partial differential equation (PDE).

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F_x}}{\partial x} + \frac{\partial \mathbf{F_x}}{\partial y} + \mathbf{H} = 0 \tag{1}$$

where $\mathbf{Q} = \{h, uh, vh\}^T$, t is time, $\mathbf{F_x} = \{uh, u^2h + \frac{gh^2}{2} - h\frac{\sigma_{xx}}{\rho}, uvh - h\frac{\sigma_{yx}}{\rho}\}^T$, $\mathbf{F_y} = \{vh, uvh - h\frac{\sigma_{xy}}{\rho}, v^2h + \frac{gh^2}{2} - h\frac{\sigma_{yy}}{\rho}\}^T$,

$\mathbf{H} = \{0, gh\frac{\partial z_b}{\partial x} + gh\frac{n^2 u\sqrt{u^2+v^2}}{h^{1/3}}, gh\frac{\partial z_b}{\partial y} + gh\frac{n^2 v\sqrt{u^2+v^2}}{h^{1/3}}\}^T$, ρ is the fluid density, u and v are the flow velocity in the x- and y-direction, respectively, h is the water depth, g is the gravitational acceleration, z_b is the riverbed elevation, n is the Manning's roughness coefficient, and σ_{ij} is the Reynolds stress due to turbulence, where the first subscript indicates the direction and the second, the face on which the stress acts. The Reynolds stresses are determined by the gradient in the mean currents using the Boussinesq approach

$$\sigma_{xx} = 2\rho\nu_t\left(\frac{\partial u}{\partial x}\right), \quad \sigma_{yy} = 2\rho\nu_t\left(\frac{\partial v}{\partial y}\right), \sigma_{xy} = \sigma_{yx} = 2\rho\nu_t\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \tag{2}$$

where ν_t is the kinematic eddy viscosity, which varies spatially.

The PDE system in (1) can be solved using the finite element method with the approach of Petrov-Galerkin that incorporates a combination of Galerkin test function and a non-Galerkin component to control oscillations caused by convection [18]. The variational forms of (1) in integral form and in finite element form are given as follows:

$$\int_\Omega \left[\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F_x}}{\partial x} + \frac{\partial \mathbf{F_y}}{\partial y} + \mathbf{H}\right] d\Omega = 0, \tag{3}$$

$$\sum_e \left\{\int_{\Omega_e} \Psi_i \cdot \left[\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F_x}}{\partial x} + \frac{\partial \mathbf{F_y}}{\partial y} + \mathbf{H}\right] d\Omega_e\right\} = 0, \tag{4}$$

where Ω is the domain, the subscript e indicates a particular element, the subscript i indicates a particular test function, and Ψ_i is the test function. The weak form of the equation is

$$\sum_e \left\{\begin{array}{l} \int_{\Omega_e} \left(\Psi_i \frac{\partial \mathbf{Q}}{\partial t} - \frac{\partial \Psi_i}{\partial x}\mathbf{F_x} - \frac{\partial \Psi_i}{\partial y}\mathbf{F_y} + \right. \\ \qquad \left. \varphi_i A\frac{\partial \mathbf{Q}}{\partial x} B\frac{\partial \mathbf{Q}}{\partial x} + \Psi_i \mathbf{H}\right) d\Omega_e + \\ \int_{\Gamma_e} \left[\phi_i \left(\mathbf{F_x} n_x + \mathbf{F_y} n_y\right)\right] d\Gamma_e \end{array}\right\} = 0, \tag{5}$$

where ϕ_i is the Galerkin test function, φ_i is the non-Galerkin part of test function, i.e., $\beta \cdot \left[\Delta x\frac{\partial \phi_i}{\partial x}\hat{A} + \Delta y\frac{\partial \phi_i}{\partial y}\hat{B}\right]$, Δx and Δy are the grid interval in the x- and y-direction, respectively, $A = \frac{\partial \mathbf{F_x}}{\partial \mathbf{Q}}$, $B = \frac{\partial \mathbf{F_y}}{\partial \mathbf{Q}}$, Γ_e is the boundary, and n_x and n_y are, respectively, the x- and y-component of the unit outward vector normal to the boundary. A description of $\hat{A}$ and $\hat{B}$ can be found in [19]. The temporal derivatives are expressed using a finite difference expression as

$$\left(\frac{\partial \mathbf{Q}}{\partial t}\right)_j^{k+1} = \alpha \cdot \left[\frac{\left(\frac{3}{2}\mathbf{Q}^{k+1} - \frac{1}{2}\mathbf{Q}^k\right) - \left(\frac{3}{2}\mathbf{Q}^k - \frac{1}{2}\mathbf{Q}^{k-1}\right)}{\delta t_k}\right]_j + \quad (1-\alpha) \cdot \left[\frac{\mathbf{Q}^{k+1} - \mathbf{Q}^k}{\delta t_k}\right]_j, \tag{6}$$

where the subscript j is the node index, k is the time-step index, and α is the temporal relaxation factor. A complete description of the time-stepping methods employed in ADH can be found in [20].

3. Mathematical Formulation for Mass-Conservative Unrefined Element

In this section, we present the mathematical formulation of the mass-conservative unrefinement algorithm. The derivation is based on the backward Euler time scheme. The time derivative term in (4) can be formulated using the Galerkin finite element method based on a set of linear basis functions spanning a finite dimensional space. Selecting a test function $\Psi = \sum_{i=1}^{N} p_i \Psi_i$ and finding a set of q_i for the trial function, we can approximate $\tilde{\mathbf{Q}} = \sum_{i=1}^{N} q_i \mathbf{Q_i}$ and then make

$$\int_\Omega \frac{\partial \tilde{\mathbf{Q}}}{\partial t} \Psi d\Omega = 0. \tag{7}$$

For AMR problems, the mesh may vary from time to time resulting in the set of basis functions $q^n \in \Omega^n$ and $q^{n+1} \in \Omega^{n+1}$ at time t^n and t^{n+1}, respectively. The associated finite dimensional space functions are denoted by V_h^n and V_h^{n+1} at two consecutive time-steps. Given $\mathbf{Q}_h^n \in V_h^n$, we need to find $\mathbf{Q}_h^{n+1} \in V_h^{n+1}$ such that the backward Euler time difference of (7) becomes

$$\int_{\Omega^{n+1}} \mathbf{Q}_h{}^{n+1}\Psi_h{}^{n+1}d\Omega - \int_{\Omega^{n+1}} \mathbf{Q}_h{}^{n}\Psi_h{}^{n+1}d\Omega = 0 \tag{8}$$

for all $\Psi_h{}^{n+1} \in V_h{}^{n+1}$. The second term

$$\int_{\Omega^{n+1}} \mathbf{Q}_h{}^{n}\Psi_h{}^{n+1}d\Omega \tag{9}$$

introduces problems. The function $\mathbf{Q}_h{}^n$ is defined on the mesh at previous time t^n, but the test function is defined on the domain Ω^{n+1} at time t^{n+1}. In the case of unrefinement, $V_h{}^{n+1} \subset V_h{}^n$ and $\mathbf{Q}_h{}^n$ cannot be expressed on Ω^{n+1} exactly.

3.1. Mass-conservative merged element using local approach

The goal of the mass conservation scheme is to preserve the mass in a merged element, in which a node has been removed. In Figure 1, elements e_1 and e_2 are merged to element e^* due to unrefining the node (x_4, y_4). The local approach to accurately calculate (9) allows a temporary and possibly discontinuous solution $\mathbf{Q}^*$ for the merged element only. Therefore, compute the $\{\mathbf{Q}_1^*, \mathbf{Q}_2^*, \mathbf{Q}_3^*\}$ values using the basis functions $\{N\}$ such that

$$\begin{aligned}\int_{\Omega_{e^*}} \left(\mathbf{Q}_1{}^* N_1{}^{n+1} + \mathbf{Q}_2{}^* N_2{}^{n+1} + \mathbf{Q}_3{}^* N_3{}^{n+1}\right)\Psi_i{}^{n+1}dA = \\ \int_{\Omega_{e_1}} \left(\mathbf{Q}_1{}^n N_1{}^n + \mathbf{Q}_4{}^n N_2{}^n + \mathbf{Q}_3{}^n N_3{}^n\right)\Psi_i{}^{n+1}dA + \\ \int_{\Omega_{e_2}} \left(\mathbf{Q}_4{}^n N_1{}^n + \mathbf{Q}_2{}^n N_2{}^n + \mathbf{Q}_3{}^n N_3{}^n\right)\Psi_i{}^{n+1}dA\end{aligned} \tag{10}$$

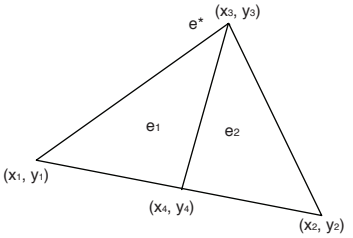


Figure 1: Element with a unrefined node

3.2. Local linear system to solve

3.2.1. LHS

On the left-hand side of (10), the formulation can be rewritten as shown in (11), and the integration is as shown as (12), where A_{e^*} is the area of element e^*.

$$LHS = \int_{\Omega_{e^*}} \begin{bmatrix} N_1{}^{n+1} \\ N_2{}^{n+1} \\ N_3{}^{n+1} \end{bmatrix} \begin{bmatrix} N_1{}^{n+1} & N_2{}^{n+1} & N_3{}^{n+1} \end{bmatrix} dA \begin{Bmatrix} \mathbf{Q}_1^* \\ \mathbf{Q}_2^* \\ \mathbf{Q}_3^* \end{Bmatrix} \tag{11}$$

$$LHS = \frac{A_{e^*}}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{Bmatrix} \mathbf{Q}_1^* \\ \mathbf{Q}_2^* \\ \mathbf{Q}_3^* \end{Bmatrix}. \tag{12}$$

3.2.2. RHS

On the right-hand side of (10), the formulation can be rewritten to

$$\left(\int_{\Omega_{e_1}} \begin{bmatrix} N_1{}^n N_1{}^{n+1} & 0 & N_3{}^n N_1{}^{n+1} & N_2{}^n N_1{}^{n+1} \\ N_1{}^n N_2{}^{n+1} & 0 & N_3{}^n N_2{}^{n+1} & N_2{}^n N_2{}^{n+1} \\ N_1{}^n N_3{}^{n+1} & 0 & N_3{}^n N_3{}^{n+1} & N_2{}^n N_3{}^{n+1} \end{bmatrix} dA + \int_{\Omega_{e_2}} \begin{bmatrix} 0 & N_2{}^n N_1{}^{n+1} & N_3{}^n N_1{}^{n+1} & N_1{}^n N_1{}^{n+1} \\ 0 & N_2{}^n N_2{}^{n+1} & N_3{}^n N_2{}^{n+1} & N_1{}^n N_2{}^{n+1} \\ 0 & N_2{}^n N_3{}^{n+1} & N_3{}^n N_3{}^{n+1} & N_1{}^n N_3{}^{n+1} \end{bmatrix} dA \right) \{\mathbf{Q}\}, \tag{13}$$

where $\{\mathbf{Q}\} = [\mathbf{Q}_1^n\ \mathbf{Q}_2^n\ \mathbf{Q}_3^n\ \mathbf{Q}_4^n]^T$.

Once the mathematical form of (13) is obtained, the solution of $\{\mathbf{Q}^*\}$ can then be solved using a direct solver from Equations (12) and (13).

4. Software implementation

The data structure is designed as in the following box on the left to store the merged element number, the edge number that the unrefined node sits on, the unrefined nodal coordinate, the quantities stored on the unrefined node, and the $\{\mathbf{Q}^*\}$ values solved by the UNR algorithm described in the previous section. Each merged element needs 152 bytes of memory to store information for computation in order to conserve water mass for flow simulations.

The following box on the right depicts the implementation of solving the shallow-water equations with AMR and mass-conservative unrefinement algorithm implementation. The function `adpt_main` is the main driver for AMR comprising (1) mesh unrefinement (`adpt_unref`), (2) sweeps of mesh refinement (`adpt_ref`), and (3) repartition if needed. At the end of each of these two functions, `node_renumber` and `elem_renumber` are called to renumber nodes and elements. The `UNREF_elem` data structure starts to fill in for each merged element in the function `adpt_unref`, though `ustar` is computed later in the function `UNRcompute`. Note that element merge is not allowed if this is a partial wet and dry element. If the merged element is split again in the function `adpt_ref`, the data structure for this element needs to deactivate by setting the field `ielem` to -1. The data structure also needs to keep up with the new element number assigned in `elem_renumber`. If the merged element is sent to another processor, the `UNREF_elem` data must be packed for the send and unpacked after the receive on the receiving processor. The algorithm for the mass-conservative unrefined element described in the previous section is implemented in `UNRcompute`.

```
typedef struct __UNREF_elem{
   int  ielem;       /** elem number **/
   int  edge_num;    /** edge num that the
                     unrefined node sits on **/
   VECT unref_node; /** unrefined nodal
                     coordinate **/
   double *unref_value;    /** 1 + 3 +
                           ntransport **/
   double *(ustar[NDPRFC]);  /** Ditto **/
} UNREF_elem;
```

```
/* read the input file */
init_I/O
/* init of computation */
init_compute
/* partition the domain */
init_partition
/* init the computation of total mass */
init_volMass
/* renumber nodes and elements */
node_renumber
elem_renumber
/* time loop */
Foreach time-step do
  /* nonlinear loop */
  Foreach nonlinear iteration
     Solve to obtain Q values
  Endfor
  /* check mass balance of the solution */
  compute_volMass
  /* mesh adaption including unref, ref,
     partition */
  adpt_main
  /* UNR computation to obtain Q* values*/
  UNRcompute
Endfor
```

5. Experimental Results

Two ADH models, with and without a mass-conservative unrefinement algorithm (UNR), were built for investigating mass errors. The results of the volume mass obtained from running applications using these two models are compared at each simulation time-step. Two 2-D test cases are selected: (1) tidal flow with only water flow and (2) a closed pond system with water flow and conservative constituent transport.

5.1. Tidal flow test

Figure 2 shows the initial unstructured mesh containing 1843 nodes, each with three degrees of freedom described in (1), and 3424 triangular elements. Water flowed in at the top and out through the left. The right side of the bottom boundary is a dead end. The time-step size is 600 seconds throughout the entire simulation for 240 hours. The convergence tolerance for nonlinear iterations is set to 10^{-6}, and the maximum number of nonlinear iterations is set to 10.

We first run the application with no refinement. The initial volume mass is $8.560000000000000 \times 10^8$ m^3. The water mass error is almost close to zero at each simulation time-step as expected. The maximum error is

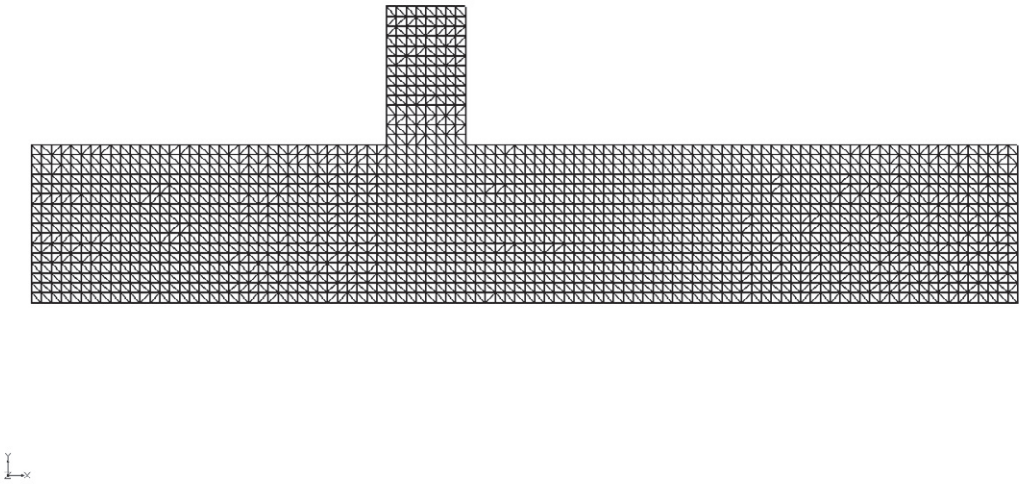


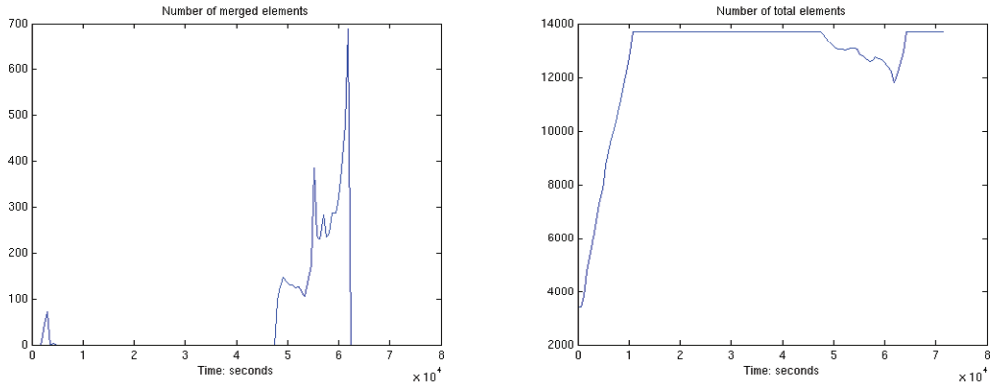
Figure 2: Tidal flow test domain and its discretized unstructured mesh

$5.125999450683594 \times 10^{-6}$ m^3, which is close to the machine error, i.e., the order of 10^{-6} out of hundreds million. We then run the same case but with adaption. The refinement tolerance (ϵ) is 1 m^3/s, and the unrefinement tolerance is 0.001 m^3/s. At each time-step, the model performs adaptive refinement if the error indicator is above 1.0 m3/s and unrefinement if the error indicator is below 0.001m3/s. Figures 3(a) and 3(b) show the number of merged elements and number of total elements at each time-step. They are changed only at the early and late stages of simulation. Figure 4 plots water mass error from the runs with or without the UNR implementation. The mass error without the UNR algorithm ranges from -48.2196 m^3 to 101.2599 m^3, while with the UNR algorithm, it ranges from -7.1525×10^{-6} m^3 to 8.9406×10^{-6} m^3. From the result, the UNR algorithm does sustain the same order of mass error from the no-adaption case. We can also observe that the more merged the elements, the worse the mass error is if UNR is not implemented. That also explains why the same mass error is maintained as the simulation approaches to the end. Note that the setup of this test is an easy physics problem to solve so that the mass error is small even without mesh adaption.

5.2. Pond test

The pond shown in Fig. 5 contains 1408 nodes, each with three degrees of freedom described in (1) and 2664 triangles initially. No inflow and outflow occur across the closed system. In addition to the flow simulation, one chemical constituent is involved in the transport process. The time-step size is 10 seconds for the entire simulation of 6000 seconds. The convergence tolerance for nonlinear iterations is 10^{-8}, and the maximum number of nonlinear iterations is set to 10.

The initial water volume is $5.4298776489890 \times 10^6$ m^3. With AMR turned off, Figures 6(a) and 6(b) display the water mass error and the chemical mass error vs simulation time. The water mass error ranges from $-2.14204192 \times 10^{-8}$ m^3 to $2.51457095 \times 10^{-8}$ m^3, while the chemical mass error ranges from $-3.35276136 \times 10^{-8}$ m^3 to $3.35276126 \times 10^{-8}$ m^3. With mesh adaption turned on, the simulations are run with and without the UNR algorithm. For both runs, the refinement tolerance is 1 m^3/s, and unrefinement tolerance is 0.001 m^3/s. Figure 7 shows the number of total elements and the number of merged elements. During the early simulation, the total number of elements increases quickly to the maximum and then gradually reduces to the initial value, while a large amount of merged elements can be found between 1000 and 3000 seconds of simulation time.



(a) Number of merged elements. (b) Number of total elements.

Figure 3: (a) and (b)

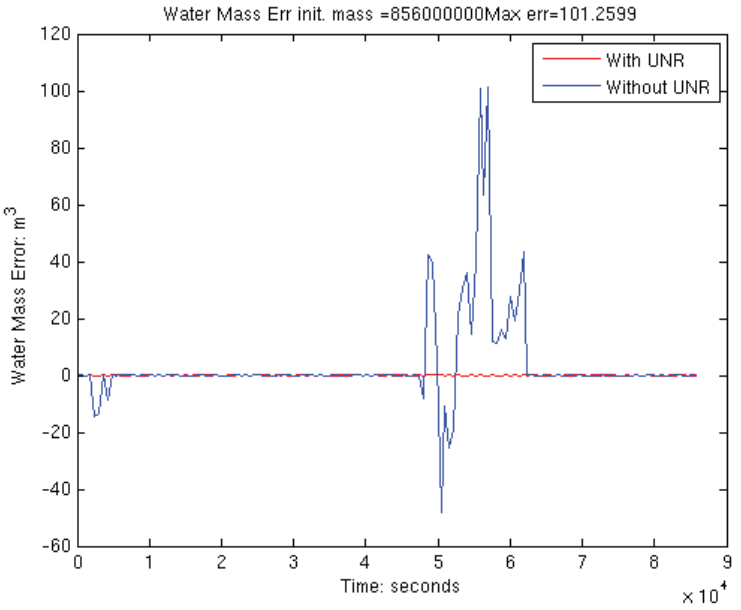


Figure 4: Water mass error with and without the UNR algorithm

Figures 8(a) and 8(b) show water mass error and chemical mass error from the run using AMR with and without the UNR algorithm. The water mass error without the UNR algorithm ranges from -14.0923 m^3 to 0.2799 m^3, while with the UNR algorithm it ranges from -3.4458×10^{-8} m^3 to 3.1664×10^{-8} m^3. For the chemical constituent, the chemical mass error without the UNR algoritm ranges -177.8483 g to 708.6960 g, while it ranges from -4.0978×10^{-8} g to 2.2383×10^{-8} g with the UNR algorithm. Therefore, the UNR algorithm can reduce the water and chemical mass error to the same order of magnitude as that obtained by running the case without adaptation. Similar to the previous test case, the pond case is an easy system to solve because the water mass error is within the range of the machine truncation error even without mesh adaption. The UNR algorithm is used to maintain the mass conservation for AMR cases, which cause mass loss or mass gain if just simply deleting a node during unrefinement.

Table 1 shows the total wall clock time in seconds vs. the number of processors. The application was run with 1, 2, 4, and 8 processors on the Cray XT4 system. The application only ran with less than 8 processors because of the limit of the small total number of elements. The result depicts the overhead of using the UNR algorithm is small.

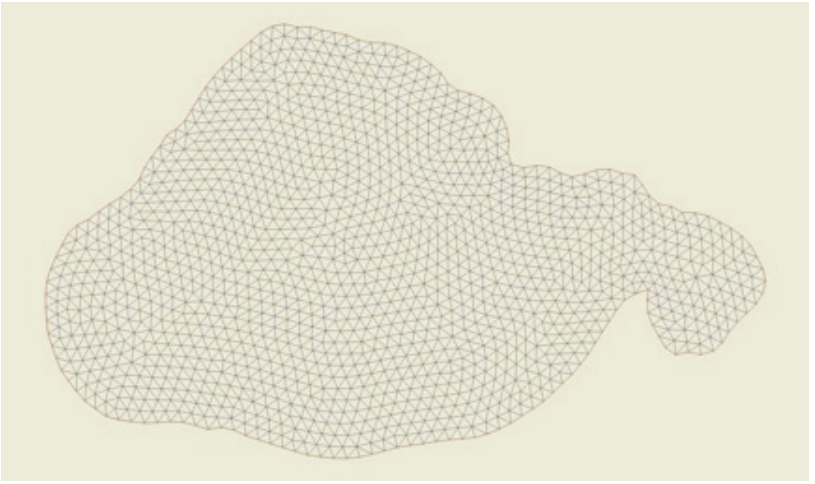
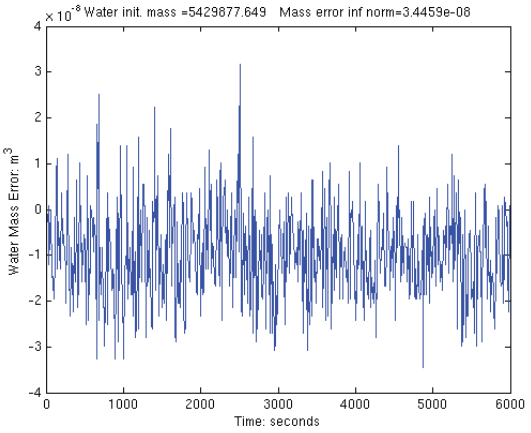
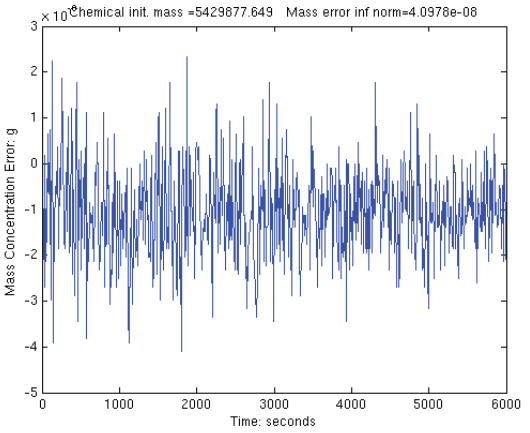


Figure 5: Domain of the pond and its initial discretized unstructured mesh



(a) Water mass error without adaptation



(b) Chemical mass error without adaptation

Figure 6: (a) and (b)

Some computational operations are executed in the UNR routine and increase the total wall clock time compared with the case without the UNR algorithm. However, the overhead resulting from the UNR algorithm is small and can be offset by the reduced number of nonlinear iterations. This is achieved by computing more accurate solutions each time.

Table 1: Total wall clock time

number of processors	Total wall clock time (secs)	
	with UNR	without UNR
1	1185.4990	1175.0420
2	677.4126	676.9971
4	351.0934	352.3615
8	177.9629	173.1149

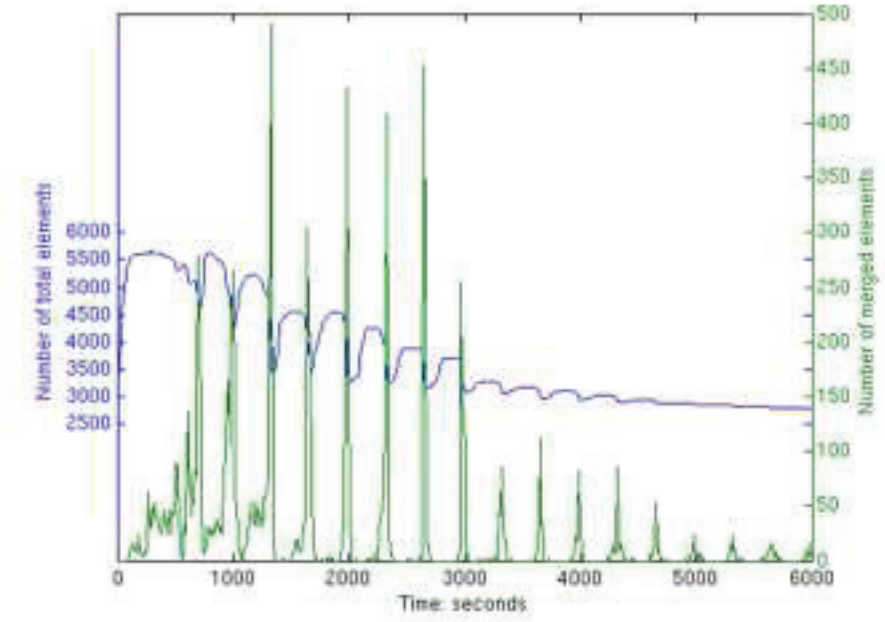
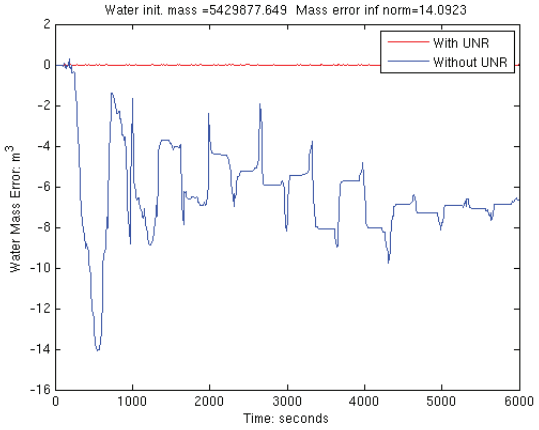
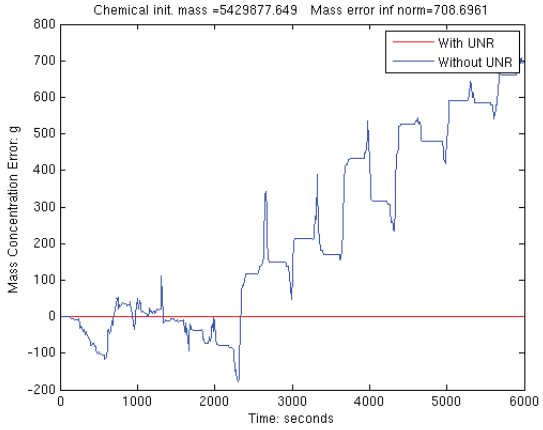


Figure 7: Total number of elements at each time from the pond case



(a) Water mass error with and without the UNR algorithm



(b) Chemical mass error with and without the UNR algorithm

Figure 8: (a) and (b)

6. Conclusions and future works

The UNR algorithm is developed and implemented successfully in the ADH model for two-dimensional shallow-water applications. It is known that there is mass error when merging elements in the common implementation of the AMR procedure. We demonstrate that the UNR algorithm can reduce water and chemical mass error to the same order of magnitude as the mass error obtained by running without mesh adaptation. In addition, the overhead incurred in the UNR algorithm is considered small. The implementation requires a small amount of memory to store and transfer information on merged elements when running on multiple processors. This research currently is presented for the backward Euler time scheme. The mass error is actually a good indicator for the setup of mesh resolution, nonlinear

tolerance, and other parameters in order to obtain accurate solutions.

We are planning to implement the UNR algorithm with the higher order time scheme in the ADH model for two-dimensional shallow-water flow. Although all the demonstration cases are small-scale problems, the extension to large scale is without limits and only requires data set generation. The application could also be extended to other modules available in ADH such as 3-D variably saturated groundwater flow. Mathematical formulation for 3-D applications will soon be discussed.

References

[1] Cheng, J.R., Cheng, H.P., Yeh, G.T.: A Lagrangian-Eulerian method with adaptively local zooming and peak/valley capturing approach to solve two-dimensional advection-diffusion transport equations. International Journal for Numerical Methods in Engineering **39** (1996) 987–1016

[2] Yeh, G.T., Cheng, J.R., Short, T.E.: An exact peak capturing and essentially oscillation-free scheme to solve advection-dispersion-reactive transport equations. Water Resources Research **28** (1992) 2937–2951

[3] Yeh, G.T., Cheng, J.R., Cheng, H.P., Sung, C.H.: An adaptive local grid refinement based on the exact peak capturing and oscillation free scheme to solve transport equations. International Journal of Computational Fluids Dynamics **24** (1995) 293–332

[4] Harten, A.: Eno schemes with subcell resolution. Journal of Computational Physics **83** (1989) 148–184

[5] Yee, H.C., Warming, R.F., Harten, A.: Implicit total variation diminishing (TVD) schemes for steady-state calculations. Journal of Computational Physics **57** (1985) 327–360

[6] Hannoun, N., Alexiades, V.: Issues in adaptive mesh refinement implementation. Electronic Journal of Differential Equations **15** (2007) 141–151

[7] Chew, L.: Guaranteed-quality triangular meshes. Technical Report TR-89–983, Computer Science Department, Cornell University (1989)

[8] Ruppert, J.: A new and simple algorithm for quality 2-dimensional mesh generation. In: Proc. 4th ACM-SIAM Symp. on Disc. Algorithms. (1993) 83–92

[9] Weatherill, N., Hassan, O., Marcum, D., Marchant, M.: Grid generation by the Delaunay triangulation. Von Karman Institute for Fluid Dynamics, 1993-1994 Lecture Series, NASA Ames Research Center (1994)

[10] Demkowicz, L., Oden, J., Rachowicz, W., Hardy, O.: Toward a universal h-p adaptive finite element strategy, part 1. Constrained approximation and data structures. Computer Methods in Applied Mechanics and Engineering **77** (1989) 79–112

[11] Oden, J., Demkowicz, L., Rachowicz, W., Westermann, T.: Toward a universal h-p adaptive finite element strategy, part 2. A posterior error estimation. Computer Methods in Applied Mechanics and Engineering **77** (1989) 113–180

[12] Rachowicz, W., Oden, J., Demkowicz, L.: Toward a universal h-p adaptive finite element strategy, part 3. Design of h-p meshes. Computer Methods in Applied Mechanics and Engineering **77** (1989) 181–212

[13] Flaherty, J.E., Paslow, P.J., Shephard, M.S., Vasilakis, J.D.: Adaptive Methods for Partial Differential Equations. Society for Industrial and Applied Mathematics, Philadelphia (1989)

[14] Jones, M.T., Plassmann, P.E.: Adaptive refinement of unstructured finite-element meshes. Finite Elements in Analysis and Design **25** (1997) 41–60

[15] Kirk, B.S., Peterson, J.W., STronger, R.H., Carey, G.F.: libmesh: a c++ library for parallel adaptive mesh refinement/ coasening simulations. Engineering with Computers **22** (2006) 237–254

[16] Yamoah, G.S.: Conservative temporal and spatial adaptive methods for groundwater flow: a dissertation. PhD thesis, Clarkson University, Potsdam, NY, USA (2009)

[17] Smith, D.S., Cerco, C.F., Berger, C.R., Savant, G., Cheng, H.P., Cheng, J.R.C., Hung, H.V., Gerald, T.K., Ai, J., Dalyander, P.A., Curtis, W.R., Davis, J.E.: Two-dimensional ADH-ICM users' manual. ERD/LAB TR-11-X, Engineer Research and Development Center, U. S. Army Corps of Engineers (2011)

[18] Berger, R.C.: HIVEL 2D v2.0 users' manual. Technical report, Waterways Experiment Station, U. S. Army Corps of Engineers (1997)

[19] Berger, R.C.: A finite element scheme for shock capturing. Technical Report HL-93-12, Waterways Experiment Station, U. S. Army Corps of Engineers (1993)

[20] Berger, R.C., Tate, J.N., Brown, G.L., Savant, G.: Adaptive hydraulics: Guidelines for solving two-dimensional shallow water problems with the adaptive hydraulics modeling system. Technical report, Engineer Research and Development Center, U. S. Army Corps of Engineers (2010)

[21] Berger, R.C., Stockstill, R.L.: Finite element model in high velocity channels. Journal of Hydraulic Engineering (1995) 710–716

[22] Berger, R.C., Tate, J.N., Brown, G.L., Savant, G.: ADaptive hydraulics quick reference: AdH version 4.01. Technical report, Engineer Research and Development Center, U. S. Army Corps of Engineers (2011) http://chl.erdc.usace.army.mil/Media/1/2/2/6/AdH_QuickReference-4.01.pdf.